

REMARKS

By this response, claims 1-4 and 7-22 are pending. Compared to prior versions, claims 1, 8, 11, 12, 16, 21 and 22 are amended while all others remain canceled or as originally or previously presented. Substantively, all claims stand rejected as anticipated by Slivka U.S. 5,943,649.

Upon inspection, Slivka indeed teaches a comparison of one or more checksums. However, Slivka's checksums relate either to code or data. In turn, code and data checksums are calculated/performed relative to code sections and data sections, respectively, of computer programs 310, 312, 314, an MS-DOS operating system 304 and/or a file management component 308. As defined, "[t]he code section of a computer program contains the computer instructions that operate upon the data of the computer program." *Col. 3, ll. 1-3*. On the other hand, "[t]he data section of the computer program contains the data that the instructions of the computer program use for operation." *Col. 3, ll. 3-5*. Further, code and data are precisely known as separate functional sections. For instance, Slivka teaches in the context of the file management component that "when performing an operation, [the file management component] always executes code as well as either reading from or writing to the data section." *Col. 3, ll. 37-39*. Accordingly, only the checksums of the code section of Slivka are relevant to the instant invention. That is, the instant invention precisely recites the validation of "executable code" and relates not to underlying data.

With this in mind, Slivka calculates a checksum for the "code section" at step 402. Later, at step 410, a "new code checksum" is calculated. In the event the first and new "code" checksums are not equivalent, corruption is indicated at step 428. Then, "processing ends, before the code is executed." *Col. 3, ll. 57-58*. Yet, this checksum comparison is wholly a one-to-one comparison of a first checksum to a second or new checksum. It is unequivocally never the calculation of "an initial score" and a calculation of subsequent

“plural scores,” let alone subsequent exclusive comparison of “each” of the subsequent plural scores to the initial score “and to no other scores,” as representatively claimed in Applicant’s independent claim 1. In other words, Slivka’s one-to-one comparison of a first checksum to a second checksum does not have multiple instances of subsequent checksums nor does it have exclusive comparisons between the multiple instances to the first checksum and to no others. As a matter of law, this feature of Slivka cannot then anticipate.

To the extent Slivka, indeed, calculates a third or more “code section” checksum (indicated in one instance by off-page connector “C” being introduced back into the flow diagram of Figure 4a before step 406), this third or more code section checksum, as well as the “new” or second code section checksum at step 410, is analyzed at step 408 to determine “whether a periodic interval has elapsed.” *Col. 3, ll. 44-45*. According to Slivka, a periodic interval “can be based on timing, count of operations requested, or other suitable events.” *Col. 3, ll. 45-46*. Purportedly, “modifications to the code section are **very rare**.” *Emphasis added, col. 3, l. 40*. In turn, it is “prefer[red]” that code section checksums not be compared or “checked upon receipt of every operation request.” *Col. 3, ll. 42-43*. In other words, Slivka believes the rarity of changes to code sections of programs, for example, enables a delay to be introduced in the code section checksum comparison routine and it is a best practice to prevent code section checksum comparisons from occurring all the time. Otherwise, processing would bog down. Expressly, the “periodic interval” routine is the mechanism for introducing delay and such is regular or periodic during code section checksum comparisons.

In great contrast, all claims of the instant invention require calculating plural subsequent scores nearly “each time” the “executable code is launched for use” after the initial loading of the executable code. Alternatively, all claims of the instant invention require calculating plural subsequent scores “randomly.” In this manner, the Applicant’s

invention contemplates thwarting hacker attempts on executable code by either being tedious or unpredictable. As indicated throughout the Applicant's Background of the Invention section of the specification, it is intensely appreciated that hackers have become extremely adept over the years and now possess exceptional talent for inserting malicious code and/or viruses into executable code. Whereas, heretofore, they were essentially unconcerned or novices at it. To wit, Slivka's teaching characterizes alterations or modifications to code as "very rare" and thus only worthy of receiving "periodic interval" checksum comparisons. The Applicant's claims, however, are not so precluded. Rather, the Applicant's invention far exceeds Slivka's contemplated hacker prevention and all claims are submitted as allowable.

Additionally, the Applicant's claims 11 and 12 precisely recite that calculations of subsequent scores are doubly performed both "randomly and substantially each time the executable code is launched for use." Nowhere does Slivka contemplate or even appreciate this sophisticated doubly secure methodology. Rather, Slivka's teaching is more naive or fledgling in approach. Further, claim 12 recites another labyrinth of protection by requiring calculation of plural subsequent scores with "predetermined time intervals" layered upon the doubly secure calculations of both "randomly and substantially each time the executable code is launched for use." Nowhere does Slivka's teaching approach this level of sophistication.

Support for the Applicant's amendments are found throughout the specification. Especially, *p. 15, ll. 9-18* recites:

[i]n this way, checks against the executable code by generating scores may be obtained every time the code is attempted to be used, periodically, or randomly. Moreover, a combination of comparisons and score generations may occur. For example, a subsequent score may be acquired each time the executable code is launched for use and then may also be acquired

randomly or periodically while the executable is in use. In this way, any alteration to the executable code, after its initial launch may be detected while the executable code is in use. As one skilled in the art will readily appreciate, this will permit changes to the image of the executable code to be detected with [sic] the executable code is executing. *Underlining added.*

Notwithstanding the above, and appreciating the Applicant's contention that Slivka's "data section" checksum teachings are beyond the scope of the claims relating to validating "executable code," to the extent Slivka calculates a first data section checksum at step 404 and compares it at step 418 to a "newly" calculated data section checksum from step 416, this is another instance of a one-to-one comparison of a first checksum to a second or new checksum. In turn, this approach cannot anticipate claims requiring multiple second scores exclusively being compared to an initial score. To the extent Slivka makes other, incremental checksums at step 426 for instance, this relates to writing operations. Subsequent checksum comparisons are then made back to the immediately one prior checksum, not the original or first data section checksum. Slivka particularly points this feature out as "the incremental checksum routine [that] adjusts the checksum **to account for a subsequent modification** by subtracting the data to be overwritten from the checksum and adding the overwriting ("new") data to the checksum." *Emphasis added, col. 4, ll. 55-59.* Stated more simply, the incremental checksum method "**recalculates** the first checksum so as to reflect the subsequent changes to the computer program after the computer program has been executed." *Emphasis added, col. 2, ll. 2-5.* In turn, this data section checksum feature of Slivka cannot, as a matter of law, anticipate claims precisely requiring "exclusive" comparisons of plural subsequent scores back to an "initial" score and "to no other scores." *See, e.g., Applicant's independent claim 1.*

The Applicant submits all claims are in a condition for allowance and requests a timely Notice of Allowance be issued for same. ***To the extent any fees are due, although***

Application No. 09/862,828
Amendment and Remarks dated January 16, 2006
Reply to Office Action dated November 3, 2005

none are believed due, the undersigned authorizes their deduction from Deposit Account No. 11-0978. Finally, the Applicant requests a change in the attorney document number of record. Namely, please replace 971-128 with 1363-004. The docket number changed when the new Power of Attorney (POA) went into effect.

Respectfully submitted,

KING & SCHICKLI, PLLC



Michael T. Sanderson
Registration No. 43,082

247 North Broadway
Lexington, Kentucky 40507
Phone: (859) 252-0889
Fax: (859) 252-0779

Certificate of Mailing

I hereby certify that this correspondence
is being deposited with the United States Postal
Service as first class mail in an envelope addressed to:
MAIL STOP AMENDMENT, Commissioner for Patents,
P.O. Box 1450, Alexandria, VA 22313-1450

on

Date

January 16, 2006
by 